Chapter 3- MICROPROCESSOR TESTABILITY.

3.1- Introduction

Microprocessors have become increasingly complex during the past several years using VLSI technology that allows hundreds of thousands of transistors to be fabricated onto a single microprocessor chip. This complexity leads to a difficult problem of how to ensure proper microprocessor operation. The number of good chips per wafer (the yield) and the number of faulty chips that escape detection during testing (the defect level) are central issues in chip design and manufacturing.

While designers of new microprocessor components generally try to save area and improve performance, customers are now asking them to include testability features that will increase product reliability. To increase reliability, manufacturers must be able to discover a high percentage of the defective chips during their testing procedures. Since chip yield is always less than 100 percent, improvements in testability are the only hope for detecting all of the defective chips in a manufactured batch.

Additional circuits must be added for testing purposes even though the ratio of active switching devices within the chip to the number of accessible input/output pins may already be very large. The difficulty of ensuring that a large chip functions properly based on controlling and observing just a few input/output pins can be enormous unless testing features are designed into the chip. An exhaustive test of all possible combinations of instruction, addressing modes, and data patterns would take millions of years to complete. In practice, functional test sets are used that often provide less than adequate fault coverage. This topic is discussed in more detail later. Manufacturers began in the early 1970's to incorporate testability techniques into their design. One of the earliest to provide specific fault coverage data on a microprocessor was Motorola with the MC6802 [DANIE85].

3.2- Testability Considerations in Microprocessor-based Design

Testability refers to the ease with which the presence and perhaps the location of a fault or faults within a system can be discovered. The key component in many such systems is a microprocessor. The entire digital system takes the form of a microcomputer comprising a microprocessor which acts as the central processing unit or system controller, ROM, RAM, and input/output circuits. Because of the complexity of these components, problems arise in testing microprocessor-based systems and in designing them to be easily testable [HAYES80]. Microprocessor-based system are difficult to test for several reasons:

- The number of possible faults is extremely large.
- Access to internal components is severely limited by the number of I/O connections available.
- A successful test will require a large number of test patterns.
- The system designer may not have a complete description of the ICs used in the unit under test.
- New and complex failure modes such as pattern sensitivity occur.

These difficulties can be greatly reduced by using design techniques specifically aimed at enhancing testability. A complete microprocessor-based system can be tested using it's microprocessor as the primary source of test pattern generation.

3.3- Microprocessor Test Generation

Among the most promising of recent test generation developments are the investigations which have attacked the test generation problem from essentially an *"architectural"* level [ROBA76], [BRAH84]. These investigators have taken a top-down approach to test

generation by extensively studying and modeling the families of microprocessors and related devices in current use. Most older test generation procedures ignore the functional identity of the various inputs, outputs, and signal paths involved in a given digital system, and simply try to generate the proper combinations of 0's and 1's to achieve certain designated goals. In contrast, these "*macro-approaches*" look directly at the architecture of the device itself and the various addresses, instructions, pointers, registers, etc., which are involved. An excellent example of this approach was pursued by Thatte and Abraham [THAT79], [ANNA82], [BRAH84], [TROY89]. They began by developing a general microprocessor model which accommodated the basic organizational features (registers, busses, ALU, etc.) of currently available microprocessors. Using this description, they then developed a "*high-level fault model*" based on the functional performance of the various basic units.

While these approaches certainly involve much more than just test generation and while considerable investigation of microprocessor test generation remains to be discussed, three testing approaches are to be considered.

3.3.1- Application-level Testing

In the case of some microprocessors used for one specific purpose, the most important point is to guarantee fault coverage of the microprocessor chip's utilized modules. Whether the unused modules function correctly is of little concern as long as the microprocessor is always used in the same way in a particular application. It is estimated that only 30-40 percent of the possible stuck-at faults in the microprocessor will be detected using this type of test [TROY89].

3.3.2- Functional Testing

Functional testing should guarantee that the chip meets system specification. Functional tests for microprocessors may be developed using several different approaches [THAT80, SRID81]. The development of a function test for a microprocessor begins with choosing a system model for describing the behavior of the chip and a fault model describing deviations from correct behavior of the system model.

One approach to modelling the system describes the data transfer behaviour of a bit slice of a microprocessor is by using a register-transfer-level (RTL) description [SRID81]. Another approach to modelling the system behaviour of a microprocessor describes the data transfer characteristics of the processor using an 'execution' graph that represents memory elements as nodes. A complete functional test for a microprocessor, using this approach, requires that the register decoding circuits, the data transfer circuits, the ALU, and the instruction execution sequencing process be tested [BRAH84]. So, for complex microprocessors with several registers and instructions, the test set can be lengthy.

Another functional testing approach can be used at the microinstruction level. Register transfer primitives can be used identified with each term in each statement in the RTL description, and the behaviour of the central processing unit (CPU) can be described by a series of these RTL statements. Test generation requires creating executable RTL statement sequences that produce output fault information when a statement executes incorrectly.

Finally, microprogramming techniques that derive a set of microprograms from user-available information are often useful. This technique involves tracing backward from

the instruction decoder output to its input to determine the set of input vectors and often requires examination of the physical layout, which is not readily available to most users.

3.3.3- Structural Testing

Structural testing examines the interconnection and operation of the primitive elements of a system. A microprocessor is considered a structure of interconnected logic gates and storage elements. Structural testing can be the most difficult to accomplish in a practical manner since much added hardware is usually required to increase the controllability and observability of buried internal signals. Most structural testing is designed to augment application and functional tests. The generation of structural tests requires access to net lists, the interconnection details of the design, and fault simulation to evaluate the quality of the test vectors. Consequently, structural testing is very rarely used on commercially available microprocessors of 100,000 transistors or more unless design for testability has been employed during its design phases to partition the machine into subunits that can be tested separately. If structural information is available for test-set generation, the expected fault coverage for this method can fall in the range of 80-90 percent, depending on the size of the subunits to be tested independently.

3.4- Testability Features of Microprocessors

Incorporating testability features into a design has been used by many manufacturers. Some important and innovative features of testability, to illustrate the use of BIST in microprocessors, are considered here. The manufacturers' increased growth in employing BIST in their real products, proves the essential features of BIST as an effective part of today's technology.

3.4.1- Motorola

The MC6802 was Motorola's first intentional attempt at testability. MC6802 was one of the earliest to provide specific fault coverage data on a microprocessor [DANIE85]. This part runs its own on-chip memory test. The experimental data resulted from tests on 93 wafers, in which 22506 dies (chips) were produced. The 22506 potential good dies were first screened for gross defects (open, shorts, etc.) prior to logic testing. Logic test 1 was the designer's test (828 clock cycles) augmented by 30 additional patterns generated from experience gained in producing the part. The fault coverage for the augmented test set was 96.6 percent. Dies were then subjected to an additional 432 vectors (logic test 2), which were designed to bring the fault coverage to 99.9 percent. The surviving dies were parametrically tested and packaged. Finally, the packaged parts were again tested yielding 4231 deliverable parts. The important data to note in this experiment is that 103 parts passed logic test 1 but failed logic test 2. Consequently, a very high-fault-coverage (> 99 percent) test-vector set is very desirable from a user's point of view to eliminate defective parts before they are inserted into more complex systems.

MC68020 [DANIE85]: Its DFT features include partitioning the machine into hardware blocks to be structurally tested (for example, PLA, ROM, RAM). The test set is built around these structural tests rather than traditional functional testing. Another DFT feature is signature analysis. An automatic test program generator (ATPG) was employed for the PLAs. The test microcode for input to the partitioned blocks is supplied by ATE from off-chip during testing.

3.4.2- Intel 80386 [GEL87], [ELAY85]

The 80386 designers incorporated test circuits into the device to ensure its testability. They designed these circuits to support proven test techniques. The 80386 uses two forms of designed-in test functions. The first is BIST, and the second is test hooks or function explicitly designed to aid testing.

The BIST feature of the machine includes three linear feedback shift registers (LFSRs) to apply exhaustive test patterns to three large PLAs. The control ROM (CROM) in the control unit has a counter on its input configured in the BIST mode to exhaustively apply all CROM addresses. Four multiple-input signature registers (MISRs) [TROY89] are used to compress the responses of the three PLAs and the CROM. The outputs of the four MISRs are again compressed in a second level of signature analysis that computes a 64-bit final result. This final 64-bit signature is compared with the "*good*" signature by the ALU. BIST is terminated when the largest PLA has sequenced through all its addresses once.

The three large PLAs and CROM are shown in their self-test form in Figure 3. Each parallel LFSR (three for the PLAs and one for the CROM) has a single-bit result that goes to a second 16-bit LFSR called the accumulation register. This is done because the PLA and CROM are embedded in the design, and the output LFSRs were not close to a major bus.

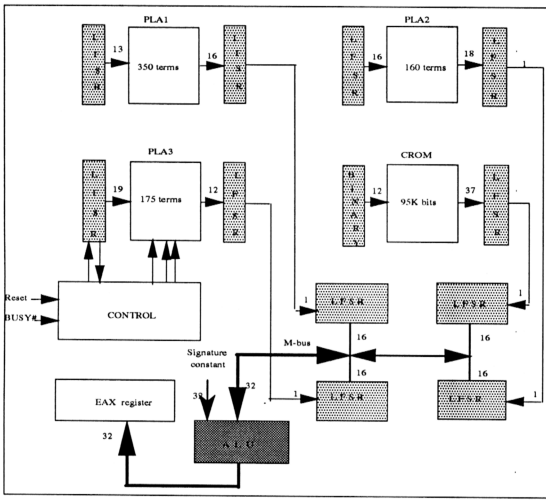

Figure 3. Intel Built-In Self-test

Each functional block with added testability feature has an area penalty ranging from 5-12%. The control PLA is an exception with a 25% area penalty caused by the extra logic needed to signal the end of BIST. Less than 2% of the overall chip area is used for testing. The test circuit covers only 36.3% of the 181000 actual transistors on the chip. Fault coverage was being computed and the expected figure for the entire test set (consisting of about a million functional vectors added to the BIST sequence) was predicted to be over 99%.

3.4.3- AT & T ASICs

AT&T has employed a partial sequential approach using circular BIST [PRAD88] in seven application-specific integrated circuits (ASICs). Four of the devices contain embedded RAM, and for all but one device, the goal was complete self-test except the I/O buffers and portions of the multiplexer logic on the inputs. The implementation uses a cell similar to that shown in Figure 4 but with the logical functions of the original BILBO. The signature readout occurs only from an LFSR or MISR embedded in the circular chain. In addition, BIST is provided for the embedded RAMs. For the six devices with full BIST, the gate count including BIST logic averages 14150 and the logic overhead averages 20%. The average fault coverage for the portion of the circuits covered by BIST for four fault-simulated circuits is 92%. Much of the hardware overhead is in the BIST cells, and a trade-off is apparent between the number of storage cells to which BIST is applied and the fault coverage. The company has automated the BIST design for standard cell VLSI implementation or PLD-based circuit packs.

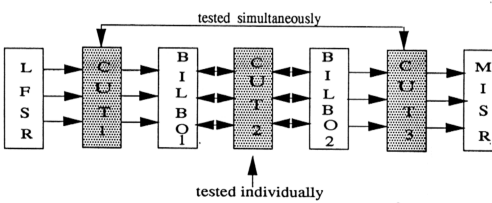

Figure 4. Self-test of AT & T ASICs.

3.4.4- IBM Micro - 370

A mixture of structural and functional testing is used in the Micro-370. The chip is divided into a control path and data path for testing purposes. Four scan register chains partition the chip into areas that are logically independent. In the data path the execution unit decoders are exhaustively tested. In the control path, most of the circuits are combinational so that structural tests may be easily derived. The state sequencer and bus controller were designed using level-sensitive scan design (LSSD) techniques [WILL83]. Standard LSSD CAD tools are also used to generate test patterns for those blocks using the LSSD methodology. Pseudorandom patterns are also used to assist the functional patterns in raising the fault coverage on some blocks.

3.4.5. TEXAS Instruments TMS370 Microprocessor [TROY89]

The TMS370 family of microprocessors is designed to use a partitioning method called parallel/serial scan design (PSSD). PSSD partitions the chip into independently testable functional modules that are reusable in other versions of the microprocessor. PSSD design rules require that the modules be test isolatable along functional boundaries to encourage module reuse. Bus architectures are encouraged since they naturally partition a chip along functional lines. Digital logic must be either RAM, ROM, combinational, shift register latch (SRL), or parallel register latches [PRL]. SRLs are used to set up scan chains. Since the PSSD rules are similar to IBM's LSSD, TESTSCAN, test coverage for the TESTSCAN vector's ranges from 83-93 percent for the module of the TMS370 family.

3.4.6. Toshiba TRON TX1 [DANI85]

Three testability methods are employed to make the TX1 testable: scan design, direct access to macroblocks, and self-test. In the TX1, all flip-flops and related registers are made scannable. The flip-flops are divided into 29 separate scan chains that operate in parallel. Some of the transistors on the chip (40%) are tested by the scan chains. Macroblocks are tested by providing external access to their control lines, by passing complicated control and decode logic. Self-test for the TX1 consists of a microinstruction controlled portion in which masked micro-orders for the machine improve the test efficiency by at least two fold. Hardware-controlled BIST is used for the PLA and micro ROM. Test coverage for the scan portion of the circuit is found to be above 90%. The remaining macroblocks are regular structures tested by microinstruction and BIST. The fault coverage for these blocks is also estimated to be above 60%.

As shown from the previous examples, BIST was adapted as a Design For Test technique. BIST methods will become more popular to either perform testing directly or enhance the controllability and observability of the devices. However, different manufacturers' techniques were used in employing BIST into the design such that a general BIST structure that is applicable to any circuit under test does not exist as yet.

3.5- Designing Testable Microprocessor

So far, the testability features for commercially available microprocessors have been briefly stated. In this section a collection of these features and other enhancements that can be incorporated to make microprocessors more testable are discussed. A microprocessor

usually consists of several regular structural blocks (PLAs, ROMs, RAMs), number of data path blocks, and some control logic blocks. The basic structural logic block types to which BIST is applied separately from general logic blocks are PLAs, ROMs and RAMs.

3.5.1. Programmable Logic Array (PLA)

PLAs are being used not only as special chips but also to realize the combinational and sequential circuits embedded in a complex VLSI chip. For instance, the Intel 80386 has three embedded PLAs and AT & T's WE32000 has eight fairly large PLAs [UPAD88]. If the PLA has only a small number of inputs (< 25), the best strategy may be exhaustive testing. Then, all we use is a pattern generation at the input, a response analyzer such as a MISR at the output (Figure 5), and a BIST control circuit to isolate the PLA from the rest of the logic. The control block initiates the self-test and at its completion evaluates the compacted results to figure out the status of the PLA. The most prevalent fault model used for PLAs are cross-point faults. Tests that detect cross-point faults also detect most other faults in PLAs. Our general assessment is that BIST will be practical only for small to medium size PLAs that can be tested exhaustively or for cases where multiple-fault detection is so important that the silicon area overhead is a secondary consideration [AGRA93]. Several BIST schemes have been proposed for PLAs [UPAD88], [TREU85].

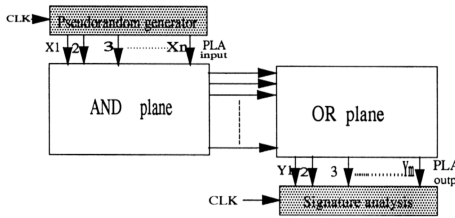

Figure 5. BIST OF PLA.

3.5.2- Read-Only Memory (ROM)

ROM is the easiest of the microprocessor modules to be tested whether it is embedded or stand-alone. The basic method is to read the contents of each word in memory and compact the outputs by means of an MISR. The resulting data at the MISR should show whether the ROM contains valid data at each address location. A description of a BIST scheme for ROMs that has very high fault coverage and very small likelihood of error escape (aliasing) was proposed by Y. Zorian, and A. Ivanor [ZORI92]. Aliasing is a problem only if the probability of multiple bit errors in the ROM outputs is high. Typical multiple bit errors are caused by the failure of an output buffer or a decoder circuit. Such faults either are not masked by the MISR or can be detected relatively easily by a few extra deterministic test vectors.

3.5.3- Read Access Memory (RAM)

RAM testing maybe done by writing a series of patterns into the memory and then reading them out to check the validity. For the application of BIST to embedded RAMs, the

pattern generator contains an address generator and a data generator as shown in Figure 6. The address generator consists of an LFSR or a counter, and the data generator is an LFSR or a finite-state machine that produces the data to be written into the RAM. During reading from the RAM, the address generator and the data can generate the same sequence of address and data again, but responses are usually compacted to protect against failure of the generators. Using the deterministic test patterns for embedded RAMs is the present trendbecause fault coverage for the deterministic method can be computed exactly for possible impacts on reliability. The sharing of test logic between multiple RAM blocks on the chip potentially reduces overhead. Thus far, most practical applications of BIST to embedded RAMs have used only algorithms that test for stuck-at faults. Among the earliest BIST RAM architecture were two proposed by Kinoshita and Saluja [SALU84]. One is based on random logic, the other on microcode. The overhead for these architectures is negligibly small (< 1%) for very long RAMs, [MEJU89, AAD90, FRAM90].

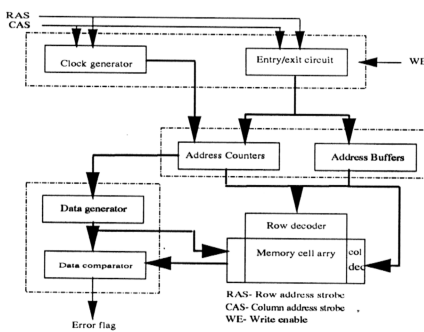

Figure 6. BIST Logic For RAM.

3.5.4- Control Unit

This usually consists of less structured finite state machine. Design approaches for testable control logics include off-line and concurrent methods. Signature analysis, data dumps, error detecting codes, boundary scans, and test-register scans may be executed as off-line. Concurrent methods include signature analysis, error-detecting codes, and hardware redundancy. Here the BIST technique proposed in [PRAD88] called circular BIST could be effective for the control logic testing. Circular BIST or circular self-test path is applicable to synchronous sequential circuits that consist of combinational logic blocks and flip-flops (finite state machine). The most favourable attribute of circular BIST is that the decrease in chip area caused by BIST circuitry, the BIST hardware overhead is compared to that of BIST techniques such as BILBO [PRAD88]. This reduces the complexity of the test supervisor and potentially reduces the test running time compared to that of other BIST techniques.

3.5.5- Data Path

The blocks used in a data path can often be tested by pseudorandom vectors. To apply pseudorandom tests pattern to the data path blocks we need to use on-chip LFSR, and a signature analyzer to compress the response. Hence, using BILBO in the microprocessor data path would be a good choice. BILBO is a LFSR-based BIST circuit with four modes of operation: normal storage, reset, serial scan, and multiple input signature analysis. By using the feedback associated with the MISR, the BILBO can also provide pattern generation as a variation of serial scan. A modified version of the BILBO structure has been proposed in a paper by Agrawal [AGRA93 part 2].

Thus, from the previous different approaches to testing a microprocessor, we can understand that there is no ideal BIST structure that would be completely general and applicable to any functional circuit. For example, a technique that works well for random combinational logic may not be suitable for a chip with RAM. By far the most common methods for built-in test pattern generation and output response analysis in designing a microprocessor are those using pseudorandom input patterns and signature analysis. Therefore, a signature analysis will continue to be the chosen technique for output response analysis and pseudorandom input pattern generation will be the dominant technique. These two techniques will be supported in the proposed design to prove their reliability and usefulness for designing a testable bit-slice microprocessor.